Week 3 : Getting started with python

pending tasks
type

Getting started with Python

Course outline:

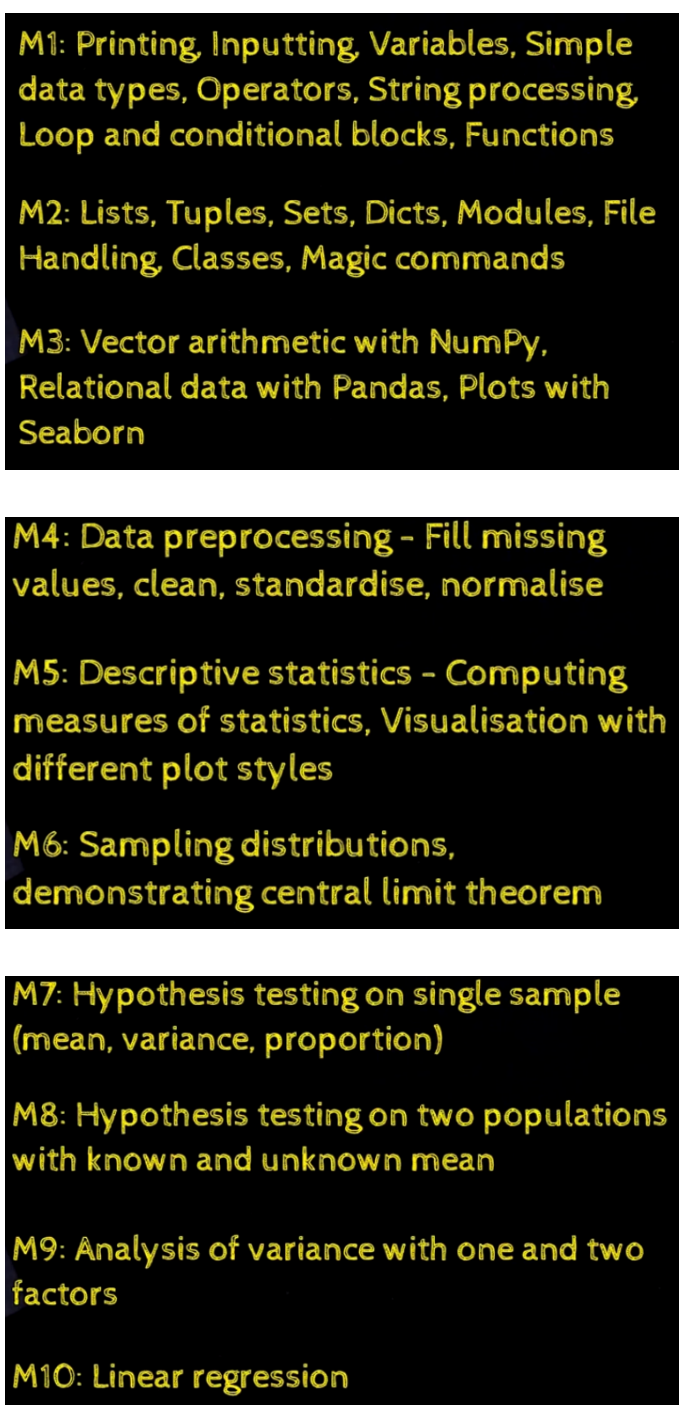

Fig.1-3 : Course outline for further concepts.

The following pdf can be used as a quick sheet for python syntax:

```
http://sixthresearcher.com/wp-content/uploads/2016/12/Python3_reference_cheat_sheet.pdf
```

Printing and basic data types

Hello world in python:

```
print('Hello world')
print('Hello' + 'world') # string concat using add operator
```

- # : used for inline comments
- any 'keyword?' will return help for the keyword.

Keyboard shortcuts:

1. cmd + enter - run a cell
2. shift + enter - run the cell and goto next cell / create new cell

Strings, integer, floating point and boolean make up the basic data types in Python.

Operators in python:

+ : performs concat for strings and addition for numeric types

/ : division for numeric types. **Return floating point.**

// : floor division for numeric types. **Returns integer.**

* : multiplication for numeric type and repetition for string.

% : reminder of numeric division.

** : exponentiation, a**n is raised to the power of n.

Boolean operators : or , and

Variables

def variables : (t = 2:24) : In python, everything is an object.

Variables store reference to python objects.

Functions on variables

```
print(type(variable)) # prints type of object variable is pointing to.
print(id(variable)) # prints memory reference of object variable points to.
```

⚠ **All the variables declared with the same value point to the same id in python.**

Integers, floating points, boolean types & input

- input(argument) : used to get input from the user, optional argument can be passed, returns a string.
- Multiple arguments can be passed to print() using comma (,) separator.
- + in print requires all arguments to be of the same type.

Type conversions:

Integer : int(value)

String : str(value)

Float : float(value)

Processing string, integer and floating point

1. The characters in string can be accessed using the index.
2. Python is zero indexed.
3. Negative indexing is used to reference the character position from the end of string.
4. Sub strings can be sliced from strings using starting and the ending index: **variable [start_index : end-index]**
5. String functions:
    - str.lower() : returns str converted to lowercase.
    - str.upper() :returns str converted to uppercase.
    - str.isupper() : returns boolean values checking if str is upper.
    - str.islower() : boolean value checking if str is lower.
    - str.find(substring) : returns position of first occurrence of the substring, -1 if not found
    - str.replace(substring_tobe_replace, replacing_string) : replaces all the occurrences of substring with given string.
6. Numeric functions:
    - var.is_integer() : returns True if var is integer else false.
    - var.as_integer_ratio() : returns a tuple of two integers that make up the fraction.

If, for, while blocks

- **Indentation is important in python.** In programming languages like C, curley braces {} are used to represent the begining and the end of a block. In python this is done using indentations.

```
# example if statement
hours_per_week = 5
if hours_per_week > 10:
    print( " you are doing well")
else:
    print(" you need to study more")

>>> you need to study more
```

- For loop is used when the **number of iterations is known**. The code below shows an example of printing fibonacci series using a for loop.

```
# fibonacci series using for loop
a = 1
b = 1
print(a)
print(b)
for i in range(10):
    temp = a + b
    a = b
    b = temp
    print(temp)
```

- While loop is used when the **termination condition is known.**

```
# fibonacci series using while loop
a = 1
b = 1
print(a)
print(b)
i = 2
while b < 1729:
    temp = a + b
    i += 1 # same as i = i + 1
    a = b
    b = temp
```

```
    print(temp)
print("Printed", i, "numbers")
```

Functions

- Write a function for component reuse.
- When a piece of code is written in multiple places, it can be encapsulated within a function and called multiple times instead. Thus, reducing redundancy.

```
def fibonacci(pos):
'''
Number at position 'pos' of fibonacci series.
'''
    a = 1
    b = 1
    for i in range(pos):
        temp = a + b
        a = b
        b = temp
    return temp
```


fun-fact : The ratio of a number and its predecessor in fibonacci series is equal to the golden ratio.

- Default values can be set foe the arguements of the function. If values are not passed during the fucntion call, then the function uses the default values.

```
# fibonacci overloaded: example of a function with default args
def fibonacci_ol(pos, a = 1, b = 1):
    for i in range(pos):
        temp = a + b
        a = b
        b = temp
    return temp
```

- A fucntion that calls itself is said to be a recursive function. Recursive functions have large call stacks, thus, not preferred at times.

```
# Example using recursive function
def fibonacci_recursive(n, a = 1, b = 1):
    if n > 1:
        return fibonacci_recursive(n - 1, b, a + b)
    else:
        return a + b
```

MCQ : Week 3

1. Which options are correct to create an empty set in Python?
    1. {}
    2. ()
    3. []
    4. **set()**
2. What is output for – max(''please help '')
    1. s
    2. A blank space
    3. e
    4. **p**

3. What is the output of the given code

```
count_var = 0
while ( count_var < 5 ):
  print( count_var  )
  pass
```

   1. prints count_var five times
   2. prints numbers 0 to 5.
   3. prints 0 five times
   4. **Infinite loop printing 0**

4. What is the output of given code

```
if( 11 == 11.0 ):
  print( 'in the if block')
else:
  print(' in the else block')
```

   1. **In the if block**
   2. In the else block
   3. Error

5. What is the output of following code:

```
a,b = b,a
```

   1. **Swaps a and b**
   2. Equates a to b and b to a , returns true if equal else returns false.
   3. Throws an error